produces such a labeling of the PUT and GET expressions of the program may be used for purposes of the present invention. In **Figure 2** statements in the program text which define a storage location X (in other words the PUTs to X in the program text) each form a node 30 in the dependence flow graph. Each statement in the program which uses memory location X (in other words expressions which GET x in the program text) forms a node 34 in the graph. A node for the aliases over the PUTs and GETs in the program text is represented by reference numeral 32. It will be seen that there is a single edge in the graph from each node 30 to node 32 and from each node 34 to node 32. In essence, the alias node 32 separates the definition-use structure of the program text.

[0016] The process of constructing a dependence flow graph such as the one shown in **Figure 2** of the drawings is illustrated in a flow chart shown in **Figure 3** of the drawings. Referring to **Figure 3**, at block 50 one node in the dependence flow graph (DFG) is associated with each PUT expression in the program. At block 52 one node in the DFG is associated with each alias in the program. At block 54 an edge is added to the DFG from the node representing each PUT expression to the node representing the alias for that put. Finally, at block 56, for each GET expression G in the right hand side of the PUT expression P, an edge is added to the dependence flow graph from the node representing the alias of G to the node representing P. A dependence flow graph constructed in accordance with the above method will have at most one edge for each PUT and GET expression in the program.

[0017] **Figure 4** of the drawings shows an algorithm, in pseudocode, for performing to perform a flow insensitive program analysis, in accordance with one embodiment of the invention. Referring to the algorithm, A is equal to the number of aliases associated with the GETs and PUTs of the program. G is a dependence flow graph defined in accordance with the above method and Q is a set of nodes of G, which is initially empty. The algorithm assigns an abstract value to each alias in the dependence flow graph. It is assumed that the abstract value from a joint complete partial order, and that for two abstract values V_1 and V_2, the expression LE (V_1, V_2) returns true if V_1 is less than or equal to V_2 in the partial order. The expression JOIN (V_1, V_2) returns the JOIN of V_1 and V_2 in the partial order. E1 is an expression in the program and Eval (E1) returns the value of E1. For each memory alias, M the expression InitialValue (M) returns an abstract value that is a safe approximation of the initial contents of the storage location (s) represented by M.

[0018] Referring to **Figure 5** of the drawings reference numeral 100 generally indicates hardware for performing term rewriting in accordance with the invention. The hardware 100 includes a memory 104, which may represent one or more physical memory devices, which may include any type of random access memory (RAM) read only memory (ROM) (which may be programmable), flash memory, non-volatile mass storage device, or a combination of such memory devices. The memory 104 is connected via a system bus 112 to a processor 102. The memory 104 includes instructions 106 which when executed by the processor 102 cause the processor to perform the methodology of the invention

as discussed above. Additionally the system 100 includes a disk drive 108 and a CD ROM drive 110 each of which is coupled to a peripheral-device and user-interface 114 via bus 112. Processor 102, memory 104, disk drive 108 and CD ROM 110 are generally known in the art. Peripheral-device and user-interface 114 provides an interface between system bus 112 and various components connected to a peripheral bus 116 as well as to user interface components, such as display, mouse and other user interface devices. A network interface 118 is coupled to peripheral bus 116 and provides network connectivity to system 100.

[0019] For the purposes of this specification, a machine-readable medium includes any mechanism that provides (i.e. stores and/or transmits) information in a form readable by a machine (e.g. computer) for example, a machine-readable medium includes read-only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other form of propagated signals (e.g. carrier waves, infra red signals, digital signals, etc.); etc.

[0020] It will be apparent from this description the aspects of the present invention may be embodied, at least partly, in software. In other embodiments, hardware circuitry may be used in combination with software instructions to implement the present invention. Thus, the techniques are not limited to any specific combination of hardware circuitry and software.

[0021] Although the present invention has been described with reference to specific exemplary embodiments, it will be evident that the various modification